Containers

Singularity and Docker (Rootless Docker)

OpenACC
More Science, Less Programming

Last updated: October 21 2022

Containers

Contents

- What is a Container?
- What is the difference between a Container and a Virtual Machine (VM)?
- Why use a container instead of a Virtual Machine?
- NGC (Nvidia GPU Cloud)
- Container Environment
  - Singularity
  - Docker (Rootless Docker)

What is a Container?

Containers are software images that holds all the needed components (code, runtime, system tools, system libraries, and software dependencies) — for an application to run easily across different computing hardware.

What is the difference between a Container and a Virtual Machine (VM)?

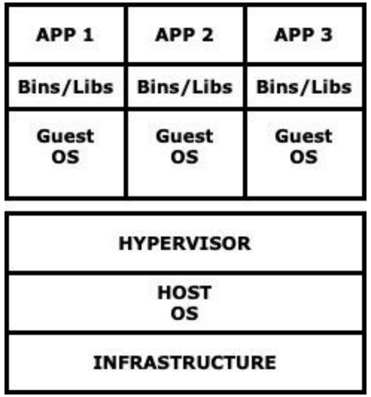
A Virtual Machine (VM) virtualizes the underlying hardware by means of a hypervisor, while it provides operating-system-level virtualization.

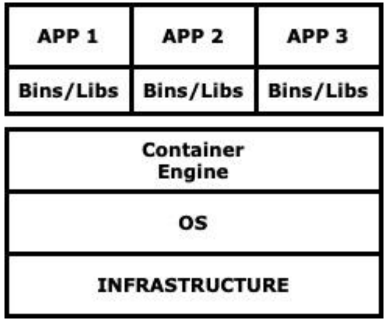
Virtual Machine

VM virtualizes the computer system.
VM sizes are comparatively large.
Examples of VMs are: KVM, Xen, VMware.

Container

Containers virtualize the operating system only.
The size of container can be smaller than the VM's.
Examples of containers are: Singularity, Docker.





"*Difference between Virtual Machines and Containers - GeeksforGeeks.*" GeeksforGeeks, 3 Jan. 2020, https://www.geeksforgeeks.org/difference-between-virtual-machines-and-containers/#:~:text=VM%20is%20piece%20of%20software,functionalities%20of%20an%20application%20independently.

Why use a Container instead of a Virtual Machine?

- Containers are more lightweight than VMs, as they are not emulating hardware.
- Containers require fewer IT resources to deploy, run, and manage.
- Containers can be created faster than VMs.
- A single system can host many more containers compared to VMs.

“Containers vs. Virtual Machines (VMS): What's the difference?” IBM. (n.d.). Retrieved September 9, 2022, from https://www.ibm.com/cloud/blog/containers-vs-vms

NVIDIA NGC Containers

OpenACC

More Science, Less Programming

Portability with NVIDIA NGC Containers

Extensive

- Diverse range of workloads and industry specific use cases

Optimized

- DL containers updated monthly
- Packed with latest features and superior performance

Secure & Reliable

- Scanned for vulnerabilities and crypto
- Tested on workstations, servers, and cloud instances

Scalable

- Supports multi-GPU and multi-node systems

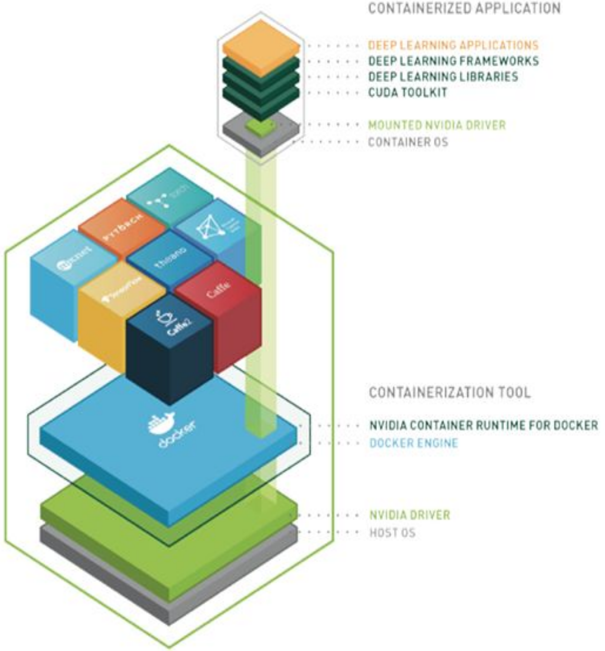
Designed for Enterprise & HPC

- Supports Docker, Singularity, and other runtimes

Run Anywhere

- Bare metal, VMs, Kubernetes, x86 etc.

NGC Deep Learning Containers



CONVERSATIONAL AI	HEALTHCARE	SMART CITIES	TELECOM	AUTONOMOUS DRIVING	ROBOTICS	HPC
JARVIS	CLARA	DEEPSTREAM & SMART PARKING	AERIAL	DRIVE	ISAAC	HPC SDK

Learn more about NVIDIA NGC Containers

Why NGC?

Optimized for Enterprise Needs

CURATED SOFTWARE

FASTER TIME TO SOLUTION

- Built and maintained by experts
- Covers popular applications and use cases
- Supercharged with latest features

SUPERIOR PERFORMANCE

RUN LARGER MODELS/SIMULATION

- AI S/W constantly optimized
- Instantly access latest features and highest performance
- Winner of MLPerf competition

TESTED ACROSS PLATFORMS

RELIABLE SOFTWARE

- Supports multi-GPU and multi-node systems
- Deploy with
- Deploy anywhere–in the cloud, on premises, and at the edges

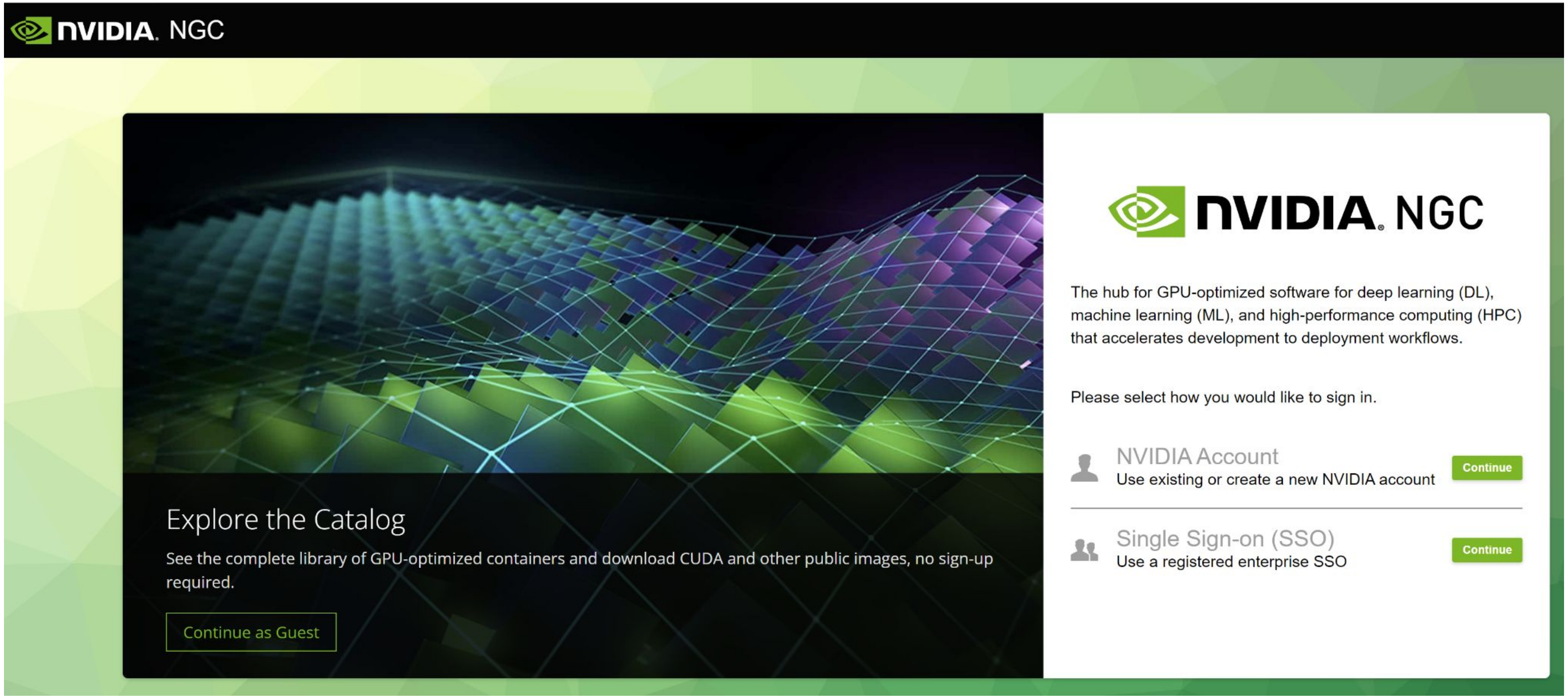
ENTERPRISE-GRADE SUPPORT

DEPLOY WITH CONFIDENCE

- Access to NVIDIA AI experts
- Minimizes system downtimes
- Security reports

Create an NGC Account and API Key

To login or create an NVIDIA NGC account, go to Sign In | NVIDIA NGC and follow the instructions.



Explore the Catalog

See the complete library of GPU-optimized containers and download CUDA and other public images, no sign-up required.

The hub for GPU-optimized software for deep learning (DL), machine learning (ML), and high-performance computing (HPC) that accelerates development to deployment workflows.

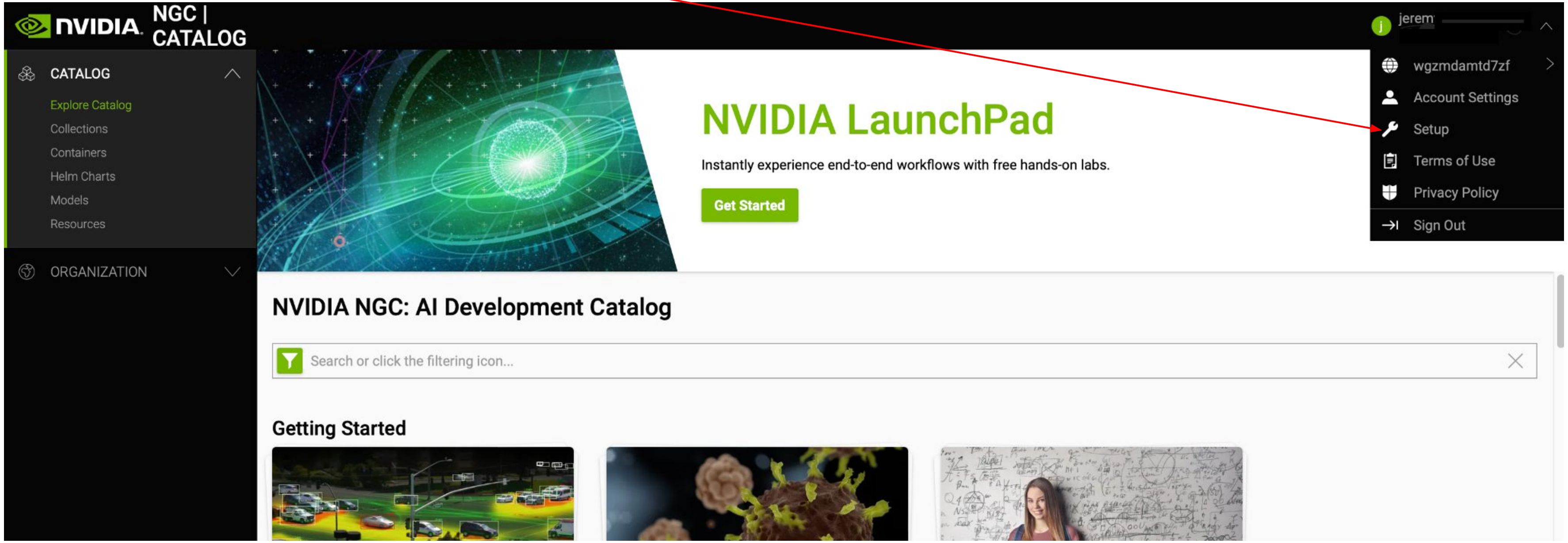
Please select how you would like to sign in.

NVIDIA Account
Use existing or create a new NVIDIA account

Single Sign-on (SSO)
Use a registered enterprise SSO

Create an NGC Account and API Key

On the NGC homescreen, select your username in the top right corner and choose Setup from the drop menu.



NVIDIA LaunchPad

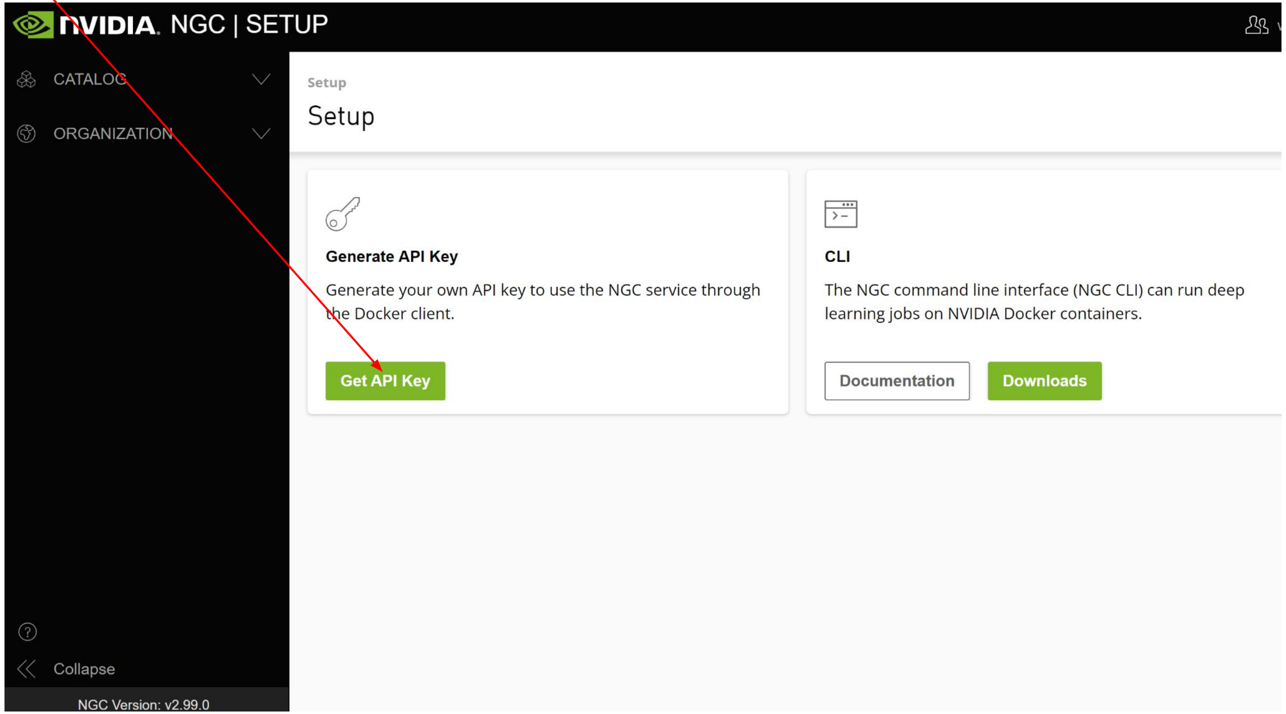
Instantly experience end-to-end workflows with free hands-on labs.

NVIDIA NGC: AI Development Catalog

Getting Started

Create an NGC Account and API Key

select Get API Key.



Setup

Generate API Key

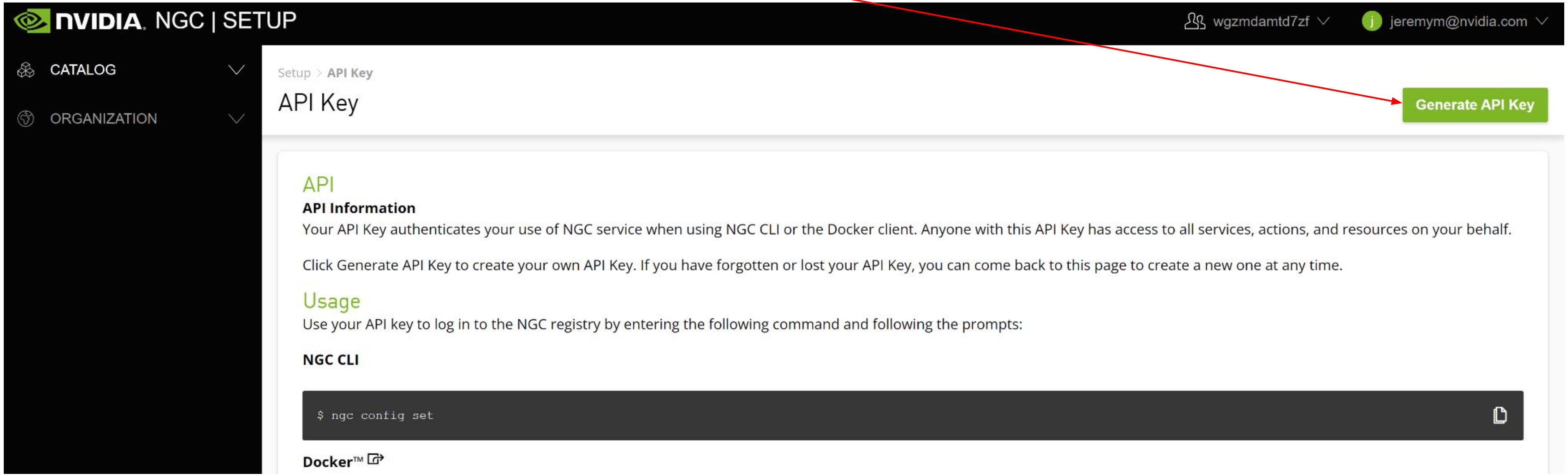
Generate your own API key to use the NGC service through the Docker client.

CLI

The NGC command line interface (NGC CLI) can run deep learning jobs on NVIDIA Docker containers.

Create an NGC Account and API Key

Click the “Generate API Key” Button



API Key

API

API Information

Your API Key authenticates your use of NGC service when using NGC CLI or the Docker client. Anyone with this API Key has access to all services, actions, and resources on your behalf.

Click Generate API Key to create your own API Key. If you have forgotten or lost your API Key, you can come back to this page to create a new one at any time.

Usage

Use your API key to log in to the NGC registry by entering the following command and following the prompts:

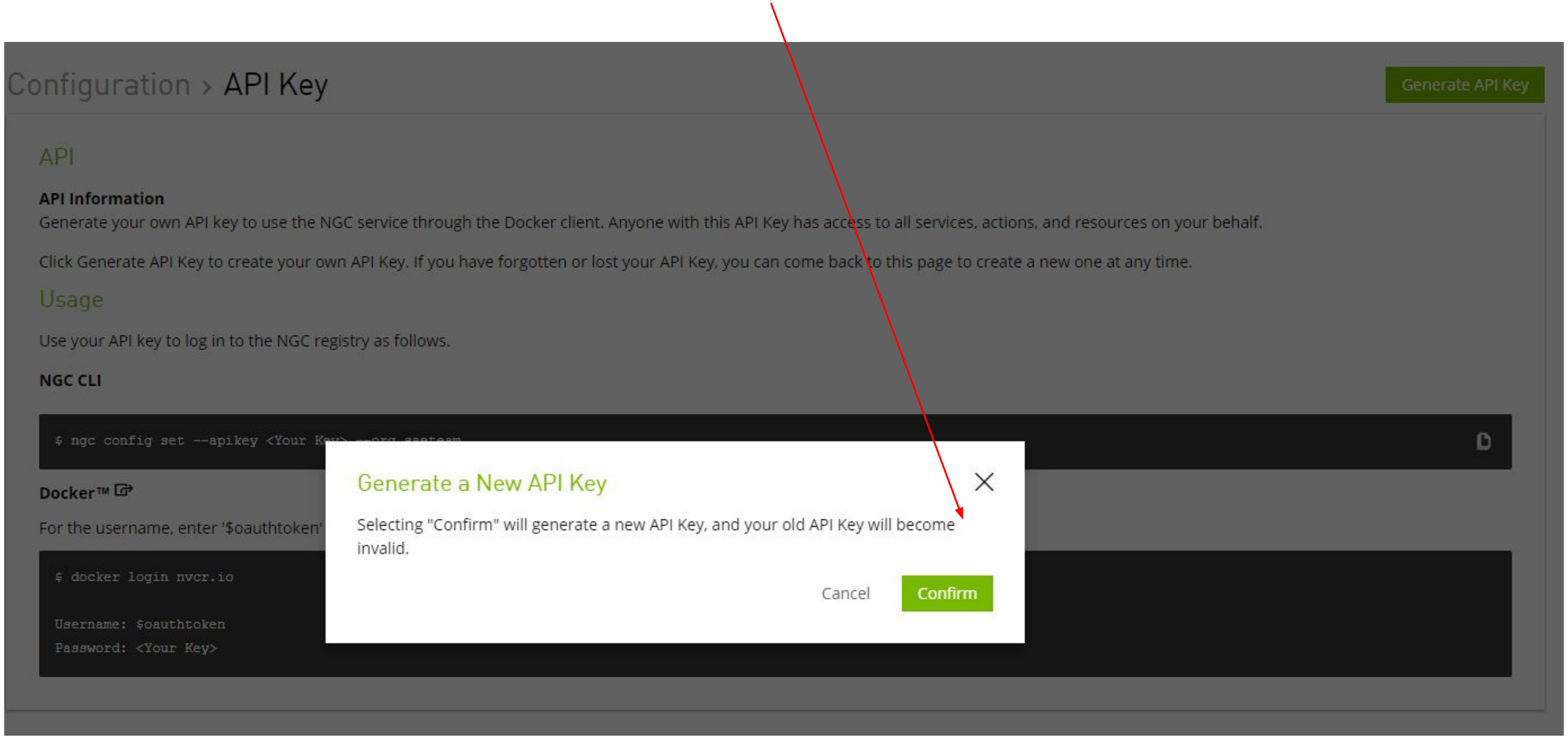
NGC CLI

```
$ ngc config set
```

Docker™

Create an NGC Account and API Key

Click the “Confirm” button to generate a new API Key



Configuration > API Key

Generate API Key

API

API Information

Generate your own API key to use the NGC service through the Docker client. Anyone with this API Key has access to all services, actions, and resources on your behalf.

Click Generate API Key to create your own API Key. If you have forgotten or lost your API Key, you can come back to this page to create a new one at any time.

Usage

Use your API key to log in to the NGC registry as follows.

NGC CLI

```
$ ngc config set --apikey <Your Key> --org saateam
```

Docker™

For the username, enter '$oauthtoken'

```
$ docker login nvcr.io

Username: $oauthtoken
Password: <Your Key>
```

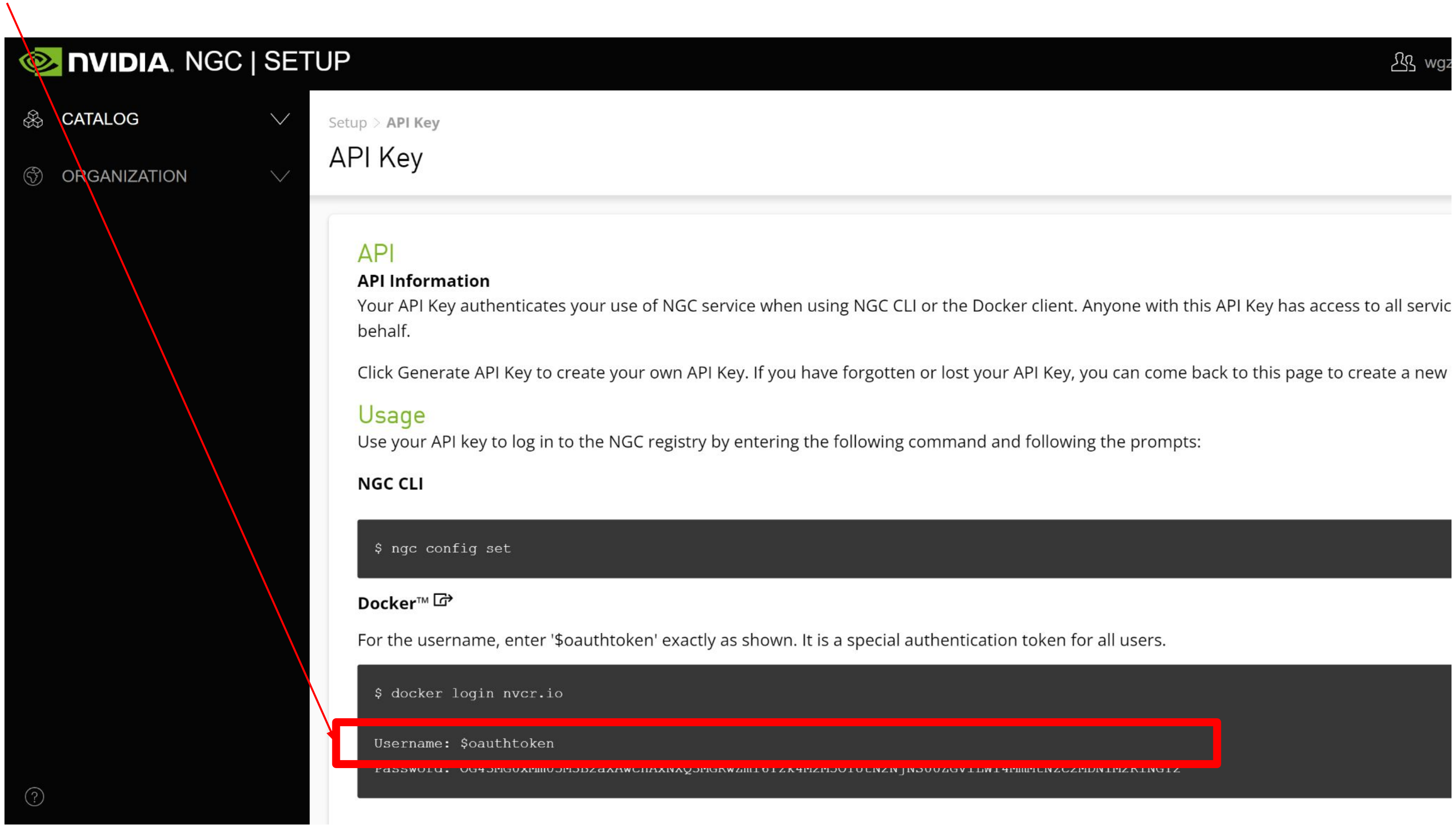
Generate a New API Key

Selecting "Confirm" will generate a new API Key, and your old API Key will become invalid.

Cancel Confirm

Create an NGC Account and API Key

Your API Key



API Key

API

API Information

Your API Key authenticates your use of NGC service when using NGC CLI or the Docker client. Anyone with this API Key has access to all servic behalf.

Click Generate API Key to create your own API Key. If you have forgotten or lost your API Key, you can come back to this page to create a new

Usage

Use your API key to log in to the NGC registry by entering the following command and following the prompts:

NGC CLI

```
$ ngc config set
```

Docker™

For the username, enter '$oauthtoken' exactly as shown. It is a special authentication token for all users.

```
$ docker login nvcr.io

Username: $oauthtoken
Password: [illegible]
```

Singularity

OpenACC

More Science, Less Programming

What is Singularity?

Singularity is a container platform. It allows you to create and run containers that bundle up pieces of software in such a way that is portable and reproducible. You can create a container with Singularity on your desktop, and then run the container on an HPC cluster, single server, in the cloud, or on a workstation. The container is a single file (simg) that is able to be run through Singularity.

- Singularity is able to run on HPC clusters (share resource systems).
- No administrator rights or sudo are needed.
- Singularity supports GPU and MPI.
- Singularity is compatible with Docker images as well as DockerHub.
- Singularity has a --fakeroot option that gives the user almost admin rights within the container.
- Singularity can build from Docker hub.

Introduction to singularity. Introduction to Singularity - Singularity container 3.5 documentation. (n.d.). Retrieved January 4, 2023, from https://docs.sylabs.io/guides/3.5/user-guide/introduction.html

Singularity Commands

Build

`build` takes a target as input and outputs a Container. Build is a multipurpose tool:

```
$ singularity [global options...] build [local options...] <IMAGE PATH> <BUILD SPEC>
```

Option1: Pull and Build a Tensorflow container from NGC with Singularity

```
$ singularity --fakeroot build tensorflow.simg docker://nvcr.io/nvidia/tensorflow:20.11-tf2-py3
```

`--fakeroot` option: fakeroot allows a user to have almost the same administrative rights as root but only within the container.

Singularity Commands

Build

Option 2: Pull and build a Singularity container from Definition Script (below) named Singularity

```
Bootstrap: docker
From: nvcr.io/nvidia/tensorflow:20.11-tf2-py3

%runscript

    "$@"

%post

    apt-get -y update
    apt-get -y install --no-install-recommends python3-pip python3-setuptools zip build-essential
    rm -rf /var/lib/apt/lists/*
    pip3 install --no-cache-dir jupyter

%files

    English/ /labs

%environment
XDG_RUNTIME_DIR=

%labels
```

```
$ singularity --fakeroot build tensorflow.simg Singularity
```

Singularity Commands

Run + Exec + Shell

- **Run**: runs the Singularity runfile

```
$ singularity run <singularity_container.simg>
```

- **Exec**: execute container commands from outside the container (like cat /etc/os-release)

```
$ singularity exec <singularity_container.simg> cat /etc/os-release
```

- **Shell**: opens a shell inside the container

```
$ singularity shell <singularity_container.simg>
```

Singularity Commands

NVIDIA GPUs

The `--nv` flag enables the container to setup the environment to use NVIDIA GPUs and the NVIDIA CUDA® libraries. Run, exec, and shell commands can take the flag.

```
$ singularity build --fakeroot tensorflow.simg docker://tensorflow/tensorflow:2.11.0rc1-gpu
```

```
$ singularity run --nv tensorflow.simg
```

Singularity Commands

Loading local folders

You can use local files from within your container

```
$ echo “hello world” > file1.txt
```

```
$ singularity exec tensorflow.simg cat file1.txt
```

```
>hello world
```

Singularity binds /home/$USER, /tmp, and $PWD at runtime, to bind other directories use:

```
$ singularity exec --bind /<host dir>:/test  tensorflow.simg ls /labs
```

This example binds the `/labs` directory to a <host dir> inside the container so you can access it, allowing you to do an `ls` on it.

Singularity Commands

Making changes to containers

Build as `--sandbox` for writable:

```
$ singularity build --fakeroot --sandbox test_tensorflow tensorflow.simg
```

Singularity Hands-On

- **Step 1**: Download Tensorflow container from docker

```
$ singularity build --fakeroot tensorflow_test.simg
docker://tensorflow/tensorflow:2.11.0rc1-gpu
```

- **Step 2**: Create sandbox from tensorflow_test.simg called tensorflow-test

```
$ singularity build --fakeroot --sandbox tensorflow-test tensorflow_test.simg
```

- **Step 3**: Run the following command

```
$ date > tensorflow-test/date
```

- **Step 4**: Build a new container (new-tensorflow.simg)

```
$ singularity build --fakeroot new-tensorflow.simg tensorflow-test/
```

- **Step 5**: Run the following command

```
$ mkdir TF-TEST ; cd TF-TEST ; echo $USER > name; cd ..
```

Singularity Hands-On

- **Step 6**: Bind folder created to the new-tensorflow.simg

```
$ singularity run --bind TF-TEST:/test new-tensorflow.simg
```

- **Step 7**: Cat the name file created earlier.

```
$ cat /test/name
```

This, if everything went correctly, will show your username

Singularity as a Slurm Job

You can run a Singularity container as a job in Slurm with the use of `srun` or `sbatch`.

Using srun to run a container:

```
$ srun --partition=gpu --gres=gpu:1 --cpus-per-gpu=4 --pty /bin/bash singularity run
--nv tensorflow.simg
```

Step 1 creates an interactive session on a compute node that has the resources that have been requested.

Step 2 starts the singularity container.

Singularity as a Slurm Job

sbatch loads a container and runs a command or script inside

```
#!/bin/bash
#SBATCH --job-name=example       # create a short name for your job
#SBATCH --partition=gpu          # what partition job will run-varies between clusters
#SBATCH --nodes=1                # node count
#SBATCH --gres=gpu:1             # selects 1 gpu
#SBATCH --ntasks=1               # total number of tasks across all nodes
#SBATCH --cpus-per-gpu=4         # cpu-cores per gpu(>1 if multi-threaded tasks)
#SBATCH --time=00:05:00          # total run time limit (HH:MM:SS)

singularity run --nv tensorflow_test.simg nvidia-smi
```

```
$ sbatch <sbatch script.sh>
```

Docker and Rootless Docker

OpenACC

More Science, Less Programming

What is Docker?

Docker is a software platform that allows you to build, test, and deploy applications quickly. Docker packages software into standardized images called containers that have everything the software needs to run including libraries, system tools, code, and runtime.

What is Rootless Docker?

Rootless Docker allows the running of Docker containers as a non-root user.

Accelerated, containerized application development. Docker. (2022, October 25). Retrieved October 4, 2022, from https://www.docker.com/

Starting Rootless Docker

Load Rootless Docker module if it is installed as a module, if not please contact your cluster administrator.

```
$ module load rootless-docker

$ start_rootless_docker.sh –quiet
```

Continue on and use Docker without sudo. (The following commands will not include sudo. If you are not using Rootless Docker, please verify that you have sudo rights.)

Docker/Rootless Docker

Useful Definitions

Docker images are the basis of containers

- An image is an ordered collection of root filesystem changes and the corresponding execution parameters for use within a container runtime.
- An image typically contains a union of layered filesystems stacked on top of each other.
- An image does not have state and it never changes.

Container

- A container is a runtime instance of a docker image.
- A Docker container consists of:
  - A Docker image
  - Execution environment
  - A standard set of instructions

Myrianthous, G. (2022, August 11). *Docker image vs Container*. Medium. Retrieved September 9, 2022, from https://towardsdatascience.com/docker-image-vs-container-7d9acab24c5

Docker/Rootless Docker

Common Commands

List Images:

```
docker images
```

Remove an Image:

```
docker rmi imageID
```

Remove all Images:

The `-a` flag means "all" and the -q flag makes the output a list of imageID's.

```
docker rmi $(docker images -a -q)
```

List Containers:

```
docker ps -a
```

Remove a Container:

```
docker rm containerID
```

Remove all Containers:

The `-f` flag will force a container shutdown

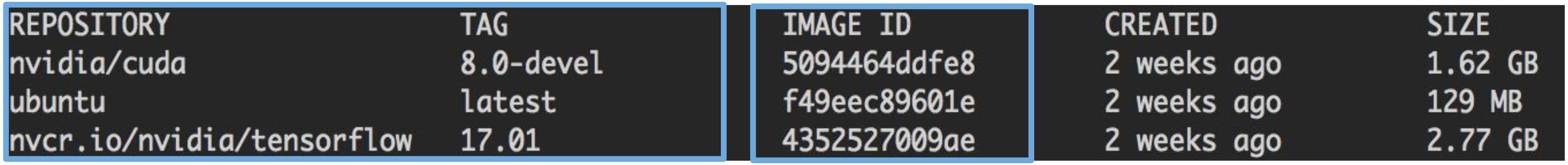
Stop a running Container:

```
docker stop containerID
```

Note: imageID and containerID can be either a hash or a name

Docker/Rootless Docker

Image Details



```
REPOSITORY                    TAG           IMAGE ID        CREATED        SIZE
nvidia/cuda                   8.0-devel     5094464ddfe8    2 weeks ago    1.62 GB
ubuntu                        latest        f49eec89601e    2 weeks ago    129 MB
nvcr.io/nvidia/tensorflow     17.01         4352527009ae    2 weeks ago    2.77 GB
```

Image Name = Repository:Tag

ImageID = Unique Hash

Docker/Rootless Docker

Running Containers

Docker run Options

- `--rm` remove the container after it exits
- `-gpus` for GPU isolation
- `-i -t` or `-it` interactive, and connect a "tty"
- `--name` give the container a name
- `-p 5004:8888` map port 8888 on the host to 5004 inside the container
- `-v ~/data:/data` map storage volume from host to container (bind mount) i.e. bind the `~data` directory in your home directory to `/data` in the container

Note: Use ***nvidia-docker run***... instead if you're using docker version<19.03.

Starts TensorFlow with ports, volumes, console, and comment (All 1 line):

```
docker run
      --rm -it
      -gpus all
      --name
      -p 5004:8888
      -v ~/data:/data
      nvcr.io/nvidia/tensorflow:20.06-py3
```

Docker Container LS. Docker Documentation. (2022, September 9). Retrieved September 9, 2022, from https://docs.docker.com/engine/reference/commandline/container_ls/

Docker/Rootless Docker

Validating Docker

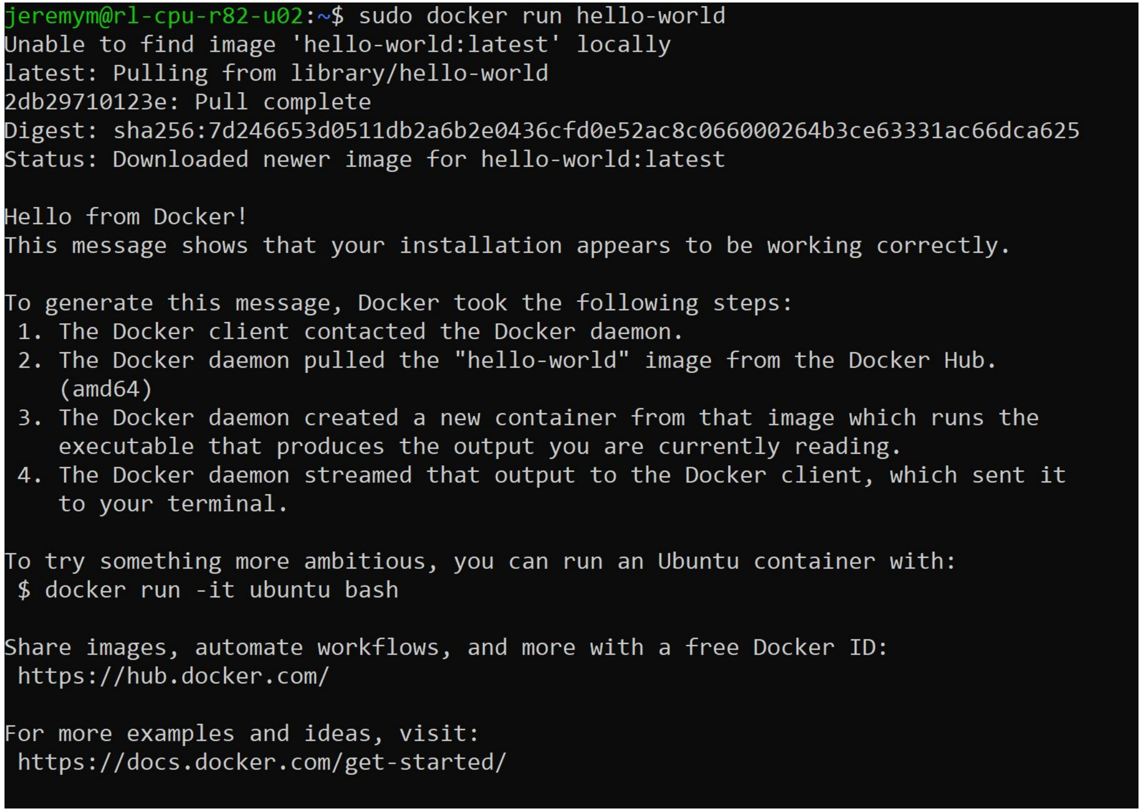
Run hello-world to test docker is up and running:
If using just Docker:

```
$ sudo docker run hello-world
```

If using Rootless Docker:

```
$ docker run hello-world
```

Output should look like..



```
jeremym@rl-cpu-r82-u02:~$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
2db29710123e: Pull complete
Digest: sha256:7d246653d0511db2a6b2e0436cfd0e52ac8c066000264b3ce63331ac66dca625
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

Docker/Rootless Docker Hands-On

If Rootless Docker is not used, you will need to use sudo or have administrator rights.

- **Step 1**: Pull Docker image

```
$ docker pull tensorflow/tensorflow:2.11.0rc1-gpu
```

- **Step 2**: Run the container and

```
$ docker run -it --name="tensor-test" tensorflow/tensorflow:2.11.0rc1-gpu
 /bin/bash
```

- **Step 3**: Run the following commands

```
$ mkdir test && mkdir test-2 && cd test
$ date > date.txt
$ exit
```

- **Step 4**: Build a new container

```
$ docker commit -m="test" tensor-test tensor-test
```

Docker/Rootless Docker Hands-On

- **Step 5**: Reopen the new container

```
$ docker run -it -v  "$(pwd)":/test2/ tensor-test /bin/bash
```

- **Step 6**: Cat the date.txt file created, should display the text in date.txt

```
$ cat /test/date.txt
```

verify that /test2 is mapped to your local home directory

- **Step 7**: Exit the container and stop the container

```
$ exit
```

```
$ docker stop tensor-test
```

Resources and Links

- Additional resources
  - Docker, Inc.
  - Sylabs Inc.
  - Open Hackathons technical resource page
  - Open Hackathons GitHub Repository

- Join the OpenACC and Hackathons Slack channel

- Licensing

Learn more at

WWW.OPENACC.ORG

OpenACC

More Science, Less Programming